Package ‘MixStable’

November 3, 2025

Type Package

Title Parameter Estimation for Stable Distributions and Their Mixtures

Version 0.1.0

Description Provides various functions for parameter estimation of one-dimensional stable distributions and their mixtures. It implements a diverse set of estimation methods, including quantile-based approaches, regression methods based on the empirical characteristic function (empirical, kernel, and recursive), and maximum likelihood estimation. For mixture models, it provides stochastic expectation–maximization (SEM) algorithms and Bayesian estimation methods using sampling and importance sampling to overcome the long burn-in period of Markov Chain Monte Carlo (MCMC) strategies. The package also includes tools and statistical tests for analyzing whether a dataset follows a stable distribution. Some of the implemented methods are described in Hajjaji, O., Manou-Abi, S. M., and Slaoui, Y. (2024) <doi:10.1080/02664763.2024.2434627>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Imports stabledist, mixtools, nortest, openxlsx, e1071, jsonlite, libstable4u, stats, graphics, MASS, utils

Suggests ggplot2, grDevices, moments, readxl, testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Solym Manou-Abi [aut, cre],
Adam Najib [aut],
Yousri Slaoui [aut]

Maintainer Solym Manou-Abi <solym.manou.abi@univ-poitiers.fr>

Repository CRAN

Date/Publication 2025-11-03 19:20:18 UTC

Contents

aic — *Akaike Information Criterion (AIC)*

Description

Computes the AIC value given a log-likelihood and number of parameters.

Usage

```
aic(log_likelihood, num_params)
```

Arguments

`log_likelihood`	Log-likelihood value.
`num_params`	Number of estimated parameters.

Value

AIC value.

analyse_stable_distribution — *Perform full stability analysis and export results*

Description

Executes the full pipeline for stability analysis: normality tests, QCV computation, parameter estimation, plotting, and report export.

Usage

```
analyse_stable_distribution(
  x,
  filename = "interval_analysis",
  qcv_threshold = 1.8,
  fig_path = NULL,
  verbose = FALSE
)
```

Arguments

`x`	Numeric vector of data.
`filename`	Base name for output files (JSON and Excel). Written to tempdir().
`qcv_threshold`	Threshold for QCV to consider distribution heavy-tailed.
`fig_path`	Optional path to save the plot (PNG). If NULL, uses tempdir().
`verbose`	Logical; if TRUE, prints progress messages. Default is FALSE.

Value

Invisibly returns a character string containing a Markdown-formatted summary report.

`bayesian_mixture_model` *Bayesian mixture model using normal components (simplified)*

Description

Bayesian mixture model using normal components (simplified)

Usage

```
bayesian_mixture_model(data, draws = 1000, chains = 2)
```

Arguments

`data`	Numeric vector of observations.
`draws`	Number of iterations per chain.
`chains`	Number of chains to simulate.

Value

List containing model fit, chains, weights, means, and standard deviations.

`bic` *Bayesian Information Criterion (BIC)*

Description

Computes the BIC value given a log-likelihood, number of parameters, and sample size.

Usage

```
bic(log_likelihood, num_params, n)
```

Arguments

`log_likelihood`	Log-likelihood value.
`num_params`	Number of estimated parameters.
`n`	Sample size.

Value

BIC value.

build_mcculloch_interpolators

Build interpolation functions from McCulloch table

Description

Constructs nearest-neighbor interpolation functions for alpha and beta based on quantile ratios (v_alpha, v_beta) from the lookup table.

Usage

```
build_mcculloch_interpolators(table)
```

Arguments

`table`	Lookup table generated by generate_mcculloch_table.

Value

List with functions interp_alpha and interp_beta.

calculate_log_likelihood

Calculate simplified log-likelihood

Description

Computes a placeholder log-likelihood value based on squared deviations from the mean. This is not a true likelihood function and is used for testing or comparison purposes.

Usage

```
calculate_log_likelihood(data, params)
```

Arguments

`data`	Numeric vector of observations.
`params`	Vector of parameters (not used in this placeholder).

Value

Scalar value representing the pseudo log-likelihood.

CDF — *Estimate stable distribution parameters using classical ECF regression*

Description

Estimate stable distribution parameters using classical ECF regression

Usage

```
CDF(x, u)
```

Arguments

`x`	Numeric vector of data.
`u`	Vector of frequency values.

Value

A list with estimated parameters: `alpha`, `beta`, `gamma`, and `delta`.

clip — *Clip values between lower and upper bounds*

Description

Clip values between lower and upper bounds

Usage

```
clip(x, lower, upper)
```

Arguments

`x`	Numeric input.
`lower`	Minimum allowed value.
`upper`	Maximum allowed value.

Value

Clipped numeric vector.

compare_em_vs_em_gibbs

Compare standard EM and EM with Gibbs sampling using kernel ECF

Description

Compares log-likelihood and weight estimates between standard EM and EM with Gibbs sampling.

Usage

```
compare_em_vs_em_gibbs(data, n_runs = 20)
```

Arguments

`data`	Numeric vector. Observed data.
`n_runs`	Integer. Number of runs for comparison.

Value

Data frame with log-likelihoods and weights for each method across runs.

compare_estimators_on_simulations

Compare MLE, ECF, and McCulloch estimators on simulated data

Description

Runs multiple simulations and compares the mean squared error (MSE) of parameter estimates from MLE, kernel ECF, and McCulloch methods.

Usage

```
compare_estimators_on_simulations(
  n_samples = 1000,
  n_runs = 30,
  interp_alpha = NULL,
  interp_beta = NULL
)
```

Arguments

`n_samples`	Integer. Number of samples per simulation.
`n_runs`	Integer. Number of simulation runs.
`interp_alpha`	Optional interpolation function for alpha (McCulloch).
`interp_beta`	Optional interpolation function for beta (McCulloch).

Value

Data frame of simulation results with MSE and estimated parameters.

---

compare_methods_across_configs
Compare McCulloch, ECF, and MLE methods across parameter configurations

---

Description

Evaluates and compares three estimation methods (McCulloch, ECF, MLE) over a set of parameter configurations.

Usage

```
compare_methods_across_configs(parameter_configs, trials = 20, n = 1000)
```

Arguments

`parameter_configs`	Named list of parameter sets (each a list with alpha, beta, gamma, delta).
`trials`	Integer. Number of trials per configuration.
`n`	Integer. Number of samples per trial.

Value

A named list of results with MSE values for each method and configuration.

---

compare_methods_with_gibbs
Compare estimation methods with and without Gibbs sampling

Description

Visualizes and compares mixture fits from EM, ECF (kernel and empirical), and Gibbs sampling.

Usage

```
compare_methods_with_gibbs(
  data,
  em_params,
  ecf_kernel_params,
  ecf_empirical_params,
  fig_dir = tempdir()
)
```

Arguments

`data`	Numeric vector of observations.
`em_params`	List of EM-estimated parameters.
`ecf_kernel_params`	List of kernel ECF-estimated parameters.
`ecf_empirical_params`	List of empirical ECF-estimated parameters.
`fig_dir`	Optional path to save plots. Defaults to tempdir().

Value

NULL (plots are saved to fig_dir)

`compute_model_metrics` *Compute log-likelihood, AIC, and BIC for alpha-stable model*

Description

Evaluates the model fit by computing the log-likelihood, AIC, and BIC for a given set of alpha-stable parameters.

Usage

```
compute_model_metrics(data, params)
```

Arguments

`data`	Numeric vector of observations.
`params`	Vector of parameters: alpha, beta, gamma, delta.

Value

List with log_likelihood, AIC, BIC, and optional error message.

`compute_quantile_ratios` *Compute McCulloch quantile ratios from sample data*

Description

This function calculates four key ratios used in McCulloch's method for estimating stable distribution parameters: - v_alpha: shape indicator - v_beta: skewness indicator - v_gamma: scale proxy - v_delta: location proxy

Usage

```
compute_quantile_ratios(X)
```

Arguments

`X`	Numeric vector of sample data.

Value

A list containing v_alpha, v_beta, v_gamma, and v_delta.

`compute_serial_interval` *Compute serial interval from CSV file*

Description

Calculates the serial interval from a CSV file containing date columns.

Usage

```
compute_serial_interval(filepath)
```

Arguments

`filepath`	Character. Path to the CSV file.

Value

Numeric vector of serial intervals.

cosine_exp_ralpha *Cosine exponential function*

Description

Cosine exponential function

Usage

```
cosine_exp_ralpha(r, x, alpha, beta, delta, omega)
```

Arguments

`r`	Integration variable.
`x`	Observation.
`alpha`	Numeric scalar. Stability parameter of the stable distribution ($0 < \alpha \leq 2$), controlling tail thickness.
`beta`	Numeric scalar. Skewness parameter of the stable distribution ($-1 \leq \beta \leq 1$).
`delta`	Numeric scalar. Location parameter of the stable distribution.
`omega`	Numeric scalar. Scale parameter of the stable distribution (omega > 0).

Value

Numeric value of the integrand.

cosine_log_weighted_exp_ralpha *Cosine-log-weighted exponential with r^(-alpha) term*

Description

Cosine-log-weighted exponential with r^(-alpha) term

Usage

```
cosine_log_weighted_exp_ralpha(r, x, alpha, beta, delta, omega)
```

Arguments

`r`	Integration variable.
`x`	Observation.
`alpha`	Numeric scalar. Stability parameter of the stable distribution ($0 < \alpha \leq 2$), controlling tail thickness.
`beta`	Numeric scalar. Skewness parameter of the stable distribution ($-1 \leq \beta \leq 1$).
`delta`	Numeric scalar. Location parameter of the stable distribution.
`omega`	Numeric scalar. Scale parameter of the stable distribution (omega > 0).

Value

Numeric value of the integrand.

`DONNEE_with_serial_interval` *Example serial interval data*

Description

A data frame containing serial interval information for transmission pairs.

Usage

```
data(DONNEE_with_serial_interval)
```

Format

A data frame with columns:

ID Integer. Unique identifier.
x.lb Date. Lower bound of infector symptom onset (YYYY-MM-DD).
x.ub Date. Upper bound of infector symptom onset (YYYY-MM-DD).
y Date. Infectee symptom onset (YYYY-MM-DD).
serial_interval Integer. Serial interval in days.

Source

Simulated or real data for demonstration.

`ecf_components` *Extract magnitude and phase components from ECF*

Description

Extract magnitude and phase components from ECF

Usage

```
ecf_components(phi_vals)
```

Arguments

`phi_vals`	Complex vector of ECF values

Value

A list with two numeric vectors: `y_vals` (log-log magnitude) and `arg_vals` (phase).

ecf_empirical — *Compute empirical characteristic function*

Description

Compute empirical characteristic function

Usage

```
ecf_empirical(X, t_grid)
```

Arguments

`X`	Numeric vector of data
`t_grid`	Vector of frequency values

Value

Complex vector of empirical characteristic function values.

ecf_estimate_all — *Estimate all stable parameters from empirical characteristic function*

Description

Estimate all stable parameters from empirical characteristic function

Usage

```
ecf_estimate_all(X, t_grid = NULL, weights = NULL)
```

Arguments

`X`	Numeric vector of data.
`t_grid`	Optional vector of frequencies (default: seq(0.1, 1.0, length.out = 50)).
`weights`	Optional vector of weights.

Value

A list containing estimated parameters: `alpha`, `beta`, `gamma`, and `delta`.

ecf_fn *Empirical Characteristic Function*

Description

Empirical Characteristic Function

Usage

```
ecf_fn(x, u, method = "simple")
```

Arguments

x	Numeric vector of data.
u	Numeric vector of frequencies.
method	Method: "simple", "kernel", or "recursive".

Value

List with magnitude and phase of ECF.

ecf_regression *Estimate stable parameters using weighted ECF regression*

Description

Estimate stable parameters using weighted ECF regression

Usage

```
ecf_regression(x, u)
```

Arguments

x	Numeric vector of data.
u	Vector of frequency values.

Value

A list with estimated parameters: `alpha`, `beta`, `gamma`, and `delta`.

`empirical_r0` *Empirical R0 estimation using growth model*

Description

Estimates the basic reproduction number (R0) using a growth model (exponential or logistic) fitted to incidence data.

Usage

```
empirical_r0(incidence, serial_interval, growth_model = "exponential")
```

Arguments

`incidence`	Numeric vector of incidence counts.
`serial_interval`	Mean serial interval.
`growth_model`	Type of growth model: "exponential" or "logistic".

Value

Estimated R0 value.

`em_alpha_stable` *EM algorithm for alpha-stable mixture*

Description

Estimates parameters of an alpha-stable mixture using EM with optional random initialization.

Usage

```
em_alpha_stable(
  data,
  n_components = 2,
  max_iter = 100,
  tol = 1e-04,
  random_init = TRUE,
  debug = TRUE
)
```

Arguments

`data`	Numeric vector of observations.
`n_components`	Integer. Number of mixture components.
`max_iter`	Integer. Maximum number of EM iterations.
`tol`	Numeric. Convergence tolerance.
`random_init`	Logical. Whether to use random initialization.
`debug`	Logical. Whether to print debug information.

Value

List with estimated weights and parameters (alpha, beta, gamma, delta).

`em_estimate_stable_from_cdf` *EM algorithm for mixture of alpha-stable distributions using CDF-based ECF*

Description

Performs EM estimation of a two-component alpha-stable mixture using a simplified empirical characteristic function derived from the cumulative distribution function (CDF).

Usage

```
em_estimate_stable_from_cdf(data, max_iter = 100, tol = 1e-04)
```

Arguments

`data`	Numeric vector of observations.
`max_iter`	Maximum number of EM iterations.
`tol`	Convergence tolerance on log-likelihood.

Value

A list with estimated component parameters (`alpha`, `beta`, `gamma`, `delta`) and mixture weight (`w`).

`em_estimate_stable_from_cdf_with_gibbs`
EM algorithm for alpha-stable mixture using CDF-based ECF and Gibbs M-step

Description

Performs EM estimation of a two-component alpha-stable mixture using a simplified empirical characteristic function derived from the cumulative distribution function (CDF), with the M-step replaced by Gibbs sampling.

Usage

```
em_estimate_stable_from_cdf_with_gibbs(data, max_iter = 100, tol = 1e-04)
```

Arguments

`data`	Numeric vector of observations.
`max_iter`	Maximum number of EM iterations.
`tol`	Convergence tolerance on log-likelihood.

Value

A list with estimated component parameters (`alpha`, `beta`, `gamma`, `delta`) and mixture weight (`w`).

`em_estimate_stable_kernel_ecf`
EM algorithm for mixture of alpha-stable distributions using kernel ECF

Description

Performs EM estimation of a two-component alpha-stable mixture using kernel-smoothed empirical characteristic function (ECF).

Usage

```
em_estimate_stable_kernel_ecf(data, max_iter = 100, tol = 1e-04)
```

Arguments

`data`	Numeric vector of observations.
`max_iter`	Maximum number of EM iterations.
`tol`	Convergence tolerance on log-likelihood.

Value

A list with estimated component parameters (`alpha`, `beta`, `gamma`, `delta`) and mixture weight (`w`).

`em_estimate_stable_kernel_ecf_with_gibbs`

EM algorithm for alpha-stable mixture using kernel ECF and Gibbs M-step

---

Description

Performs EM estimation of a two-component alpha-stable mixture using kernel-smoothed empirical characteristic function (ECF), with the M-step replaced by Gibbs sampling.

Usage

```
em_estimate_stable_kernel_ecf_with_gibbs(data, max_iter = 100, tol = 1e-04)
```

Arguments

`data`	Numeric vector of observations.
`max_iter`	Maximum number of EM iterations.
`tol`	Convergence tolerance on log-likelihood.

Value

A list with estimated component parameters (`alpha`, `beta`, `gamma`, `delta`) and mixture weight (`w`).

`em_estimate_stable_recursive_ecf`

EM algorithm for mixture of alpha-stable distributions using recursive ECF

Description

Performs Expectation-Maximization (EM) to estimate parameters of a two-component alpha-stable mixture model using recursive kernel smoothing on the empirical characteristic function.

Usage

```
em_estimate_stable_recursive_ecf(data, max_iter = 100, tol = 1e-04)
```

Arguments

`data`	Numeric vector of observations.
`max_iter`	Maximum number of EM iterations.
`tol`	Convergence tolerance on log-likelihood.

Value

A list with estimated component parameters (`alpha`, `beta`, `gamma`, `delta`) and mixture weight (`w`).

`em_estimate_stable_recursive_ecf_with_gibbs`

EM algorithm for alpha-stable mixture using recursive ECF and Gibbs M-step

---

Description

Performs EM estimation of a two-component alpha-stable mixture using recursive kernel smoothing on the empirical characteristic function (ECF), with the M-step replaced by Gibbs sampling.

Usage

```
em_estimate_stable_recursive_ecf_with_gibbs(data, max_iter = 100, tol = 1e-04)
```

Arguments

`data`	Numeric vector of observations.
`max_iter`	Maximum number of EM iterations.
`tol`	Convergence tolerance on log-likelihood.

Value

A list with estimated component parameters (`alpha`, `beta`, `gamma`, `delta`) and mixture weight (`w`).

`em_estimate_stable_weighted_ols`

EM algorithm for mixture of alpha-stable distributions using weighted OLS

Description

Performs EM estimation of a two-component alpha-stable mixture using weighted least squares regression on the ECF.

Usage

```
em_estimate_stable_weighted_ols(data, max_iter = 100, tol = 1e-04)
```

Arguments

`data`	Numeric vector of observations.
`max_iter`	Maximum number of EM iterations.
`tol`	Convergence tolerance on log-likelihood.

Value

A list with estimated component parameters (alpha, beta, gamma, delta) and mixture weight (w).

---

em_estimate_stable_weighted_ols_with_gibbs
EM algorithm for alpha-stable mixture using weighted OLS and Gibbs M-step

---

Description

Performs EM estimation of a two-component alpha-stable mixture using weighted least squares regression on the ECF, with the M-step replaced by Gibbs sampling.

Usage

```
em_estimate_stable_weighted_ols_with_gibbs(data, max_iter = 100, tol = 1e-04)
```

Arguments

data	Numeric vector of observations.
max_iter	Maximum number of EM iterations.
tol	Convergence tolerance on log-likelihood.

Value

A list with estimated component parameters (alpha, beta, gamma, delta) and mixture weight (w).

---

em_estimation_mixture *EM algorithm for two-component Gaussian mixture*

Description

Estimates parameters of a Gaussian mixture using the EM algorithm.

Usage

```
em_estimation_mixture(data, max_iter = 100, tol = 1e-06)
```

Arguments

data	Numeric vector of observations.
max_iter	Integer. Maximum number of EM iterations.
tol	Numeric. Convergence tolerance.

Value

A list containing estimated mixture parameters: pi, mu1, mu2, sigma1, sigma2.

`em_fit_alpha_stable_mixture` *EM algorithm for two-component alpha-stable mixture using MLE*

Description

Estimates parameters of a two-component alpha-stable mixture using MLE and EM.

Usage

```
em_fit_alpha_stable_mixture(
  data,
  max_iter = 200,
  tol = 1e-04,
  return_trace = FALSE
)
```

Arguments

data	Numeric vector of observations.
max_iter	Integer. Maximum number of EM iterations.
tol	Numeric. Convergence tolerance on log-likelihood.
return_trace	Logical. Whether to return trace of responsibilities and log-likelihoods.

Value

List with estimated parameters and optional trace.

`em_stable_mixture` *EM algorithm for alpha-stable mixture using a custom estimator*

Description

Performs EM estimation using a user-defined parameter estimator and ECF frequencies.

Usage

```
em_stable_mixture(data, u, estimator_func, max_iter = 300, epsilon = 0.001)
```

Arguments

data	Numeric vector of observations.
u	Numeric vector of frequency values for ECF.
estimator_func	Function to estimate stable parameters.
max_iter	Integer. Maximum number of EM iterations.
epsilon	Numeric. Convergence threshold on log-likelihood.

Value

List with estimated weights, parameters, and log-likelihood.

---

ensure_positive_scale *Ensure positive scale parameter*

Description

Ensure positive scale parameter

Usage

```
ensure_positive_scale(val, min_scale = 1e-06, max_scale = 1e+06)
```

Arguments

`val`	Numeric input.
`min_scale`	Minimum scale.
`max_scale`	Maximum scale.

Value

A finite positive scale value.

---

estimate_alpha_gamma *Estimate alpha and gamma from ECF modulus*

Description

Estimate alpha and gamma from ECF modulus

Usage

```
estimate_alpha_gamma(t_grid, phi_vals, weights = NULL)
```

Arguments

`t_grid`	Vector of frequency values.
`phi_vals`	Complex vector of ECF values.
`weights`	Optional vector of weights.

Value

A list with elements `alpha_hat` and `gamma_hat`.

estimate_beta_delta *Estimate beta and delta from ECF phase*

Description

Estimate beta and delta from ECF phase

Usage

```
estimate_beta_delta(t_grid, phi_vals, alpha_hat, gamma_hat, weights = NULL)
```

Arguments

`t_grid`	Vector of frequency values.
`phi_vals`	Complex vector of ECF values.
`alpha_hat`	Estimated alpha.
`gamma_hat`	Estimated gamma.
`weights`	Optional vector of weights.

Value

A list with elements `beta_hat` and `delta_hat`.

estimate_mixture_params *Estimate mixture of two stable distributions*

Description

Wrapper function to estimate parameters of a two-component alpha-stable mixture using MLE.

Usage

```
estimate_mixture_params(data)
```

Arguments

`data`	Numeric vector of observations.

Value

List with estimated weight and parameters for both components.

estimate_stable_from_cdf

Estimate stable parameters using CDF-based ECF regression

Description

Estimate stable parameters using CDF-based ECF regression

Usage

```
estimate_stable_from_cdf(data, u)
```

Arguments

data	Numeric vector of observations
u	Vector of frequency values

Value

A list with estimated parameters: `alpha`, `beta`, `gamma`, and `delta`.

estimate_stable_kernel_ecf

Estimate stable parameters using kernel-based ECF method

Description

Estimate stable parameters using kernel-based ECF method

Usage

```
estimate_stable_kernel_ecf(data, u)
```

Arguments

data	Numeric vector of observations
u	Vector of frequency values

Value

A list with estimated parameters: `alpha`, `beta`, `gamma`, and `delta`.

estimate_stable_params
Estimate single stable distribution parameters

Description

Wrapper function to estimate stable parameters using one of three methods: "basic", "robust", or "lbfgs".

Usage

```
estimate_stable_params(data, method = "robust")
```

Arguments

data	Numeric vector of observations.
method	Estimation method: "basic", "robust", or "lbfgs".

Value

List with estimated alpha, beta, gamma, delta.

estimate_stable_r
Estimate stable parameters using method of moments

Description

Provides a rough estimation of alpha-stable parameters using empirical moments.

Usage

```
estimate_stable_r(x)
```

Arguments

x	Numeric vector of data.

Value

List with estimated alpha, beta, gamma, delta.

`estimate_stable_recursive_ecf` *Estimate stable parameters using recursive ECF method*

Description

Estimate stable parameters using recursive ECF method

Usage

```
estimate_stable_recursive_ecf(data, u)
```

Arguments

`data`	Numeric vector of observations.
`u`	Vector of frequency values.

Value

A list with estimated parameters: `alpha`, `beta`, `gamma`, and `delta`.

`estimate_stable_weighted_ols` *Estimate stable parameters using weighted OLS on recursive ECF*

Description

Estimate stable parameters using weighted OLS on recursive ECF

Usage

```
estimate_stable_weighted_ols(data, u)
```

Arguments

`data`	Numeric vector of observations
`u`	Vector of frequency values

Value

A list with estimated parameters: `alpha`, `beta`, `gamma`, and `delta`.

est_r0_ml — *Estimate R0 using maximum likelihood*

Description

Computes the basic reproduction number (R0) using a maximum likelihood approach based on secondary cases and generation time weights.

Usage

```
est_r0_ml(W, N)
```

Arguments

`W`	Generation time distribution.
`N`	Secondary case counts.

Value

Estimated R0 value.

est_r0_mle — *MLE estimation of R0 using generation time*

Description

Estimates the basic reproduction number (R0) using maximum likelihood and a parametric model based on generation time and incidence data.

Usage

```
est_r0_mle(incidence, gen_time)
```

Arguments

`incidence`	Numeric vector of incidence counts.
`gen_time`	Mean generation time.

Value

Estimated R0 value.

eta0 *Helper function for eta0 computation*

Description

Helper function for eta0 computation

Usage

```
eta0(u, alpha, gamma, eps = 0.05)
```

Arguments

u	Frequency vector.
alpha	Stability parameter.
gamma	Scale parameter.
eps	Tolerance for $\alpha \approx 1$.

Value

Numeric vector of eta0 values.

eta_func *General eta function*

Description

General eta function

Usage

```
eta_func(t, alpha, gamma)
```

Arguments

t	Frequency vector.
alpha	Stability parameter.
gamma	Scale parameter.

Value

Numeric vector of eta values.

`evaluate_estimation_method` *Evaluate estimation method using MSE over multiple trials*

Description

Computes the mean squared error (MSE) of a parameter estimation function over several trials.

Usage

```
evaluate_estimation_method(
  estimator_fn,
  true_params,
  n = 1000,
  trials = 20,
  seed = 42
)
```

Arguments

`estimator_fn`	Function. Estimation function to evaluate.
`true_params`	List. True parameters: alpha, beta, gamma, delta.
`n`	Integer. Number of samples per trial.
`trials`	Integer. Number of trials to run.
`seed`	Integer. Random seed for reproducibility.

Value

Numeric value representing the average MSE across trials.

`evaluate_fit` *Evaluate fit quality using RMSE and log-likelihood*

Description

Computes RMSE and log-likelihood scores for different mixture models compared to histogram data.

Usage

```
evaluate_fit(data, methods, x_vals)
```

Arguments

`data`	Numeric vector of observations.
`methods`	Named list of parameter sets (each with two components and a weight).
`x_vals`	Numeric vector. Evaluation grid.

Value

Named list of scores per method (RMSE and LogLikelihood).

`export_analysis_report` *Export analysis report to JSON and Excel*

Description

Saves the results of a stability analysis to a JSON file and an Excel workbook.

Usage

```
export_analysis_report(
  data,
  stable_params,
  qcv,
  skew_kurt,
  normality,
  verdict,
  filename = "stable_report"
)
```

Arguments

`data`	Numeric vector of data.
`stable_params`	List of estimated stable parameters.
`qcv`	QCV statistic.
`skew_kurt`	List with skewness and kurtosis.
`normality`	List of normality test results.
`verdict`	Final verdict string.
`filename`	Base name for output files.

Value

Invisibly returns a list containing the paths of the exported JSON and Excel files.

false_position_update *False position method update step*

Description

Performs a single iteration of the false position (regula falsi) method for root-finding. This method approximates the root of a function by using a linear interpolation between two points where the function changes sign.

Usage

```
false_position_update(a, b_n, f_a, f_b, objective_func)
```

Arguments

a	Numeric scalar. Left bound of the interval.
b_n	Numeric scalar. Right bound of the interval (current step).
f_a	Numeric scalar. Value of the objective function evaluated at ‘a‘.
f_b	Numeric scalar. Value of the objective function evaluated at ‘b_n‘.
objective_func	Function. The objective function whose root is being sought.

Value

Numeric scalar. Updated estimate for the root after one iteration.

fast_integrate *Fast numerical integration using trapezoidal rule*

Description

Fast numerical integration using trapezoidal rule

Usage

```
fast_integrate(func, a = -6, b = 6, N = 100)
```

Arguments

func	Function to integrate.
a	Lower bound.
b	Upper bound.
N	Number of points.

Value

Approximated integral value.

fit_alpha_stable_mle *Fit Alpha-Stable Distribution using MLE (L-BFGS-B)*

Description

Estimates the parameters of a single alpha-stable distribution using maximum likelihood and the L-BFGS-B optimization method.

Usage

```
fit_alpha_stable_mle(data)
```

Arguments

data	Numeric vector of observations.

Value

Numeric vector of estimated parameters: alpha, beta, gamma, delta.

fit_mle_mixture *Fit MLE Mixture of Two Stable Distributions*

Description

Estimates parameters of a two-component alpha-stable mixture using maximum likelihood and the L-BFGS-B optimization method.

Usage

```
fit_mle_mixture(data)
```

Arguments

data	Numeric vector of observations.

Value

Numeric vector of estimated parameters: weight, alpha1, beta1, gamma1, delta1, alpha2, beta2, gamma2, delta2.

`fit_stable_ecf` *Estimate stable parameters using filtered and weighted ECF regression*

Description

Estimate stable parameters using filtered and weighted ECF regression

Usage

```
fit_stable_ecf(data, frequencies)
```

Arguments

`data`	Numeric vector of observations.
`frequencies`	Vector of frequency values.

Value

A list with estimated parameters: `alpha`, `beta`, `gamma`, and `delta`.

`generate_alpha_stable_mixture`
Generate samples from a predefined alpha-stable mixture

Description

Simulates samples from a mixture of alpha-stable distributions using specified parameters.

Usage

```
generate_alpha_stable_mixture(
  weights,
  alphas,
  betas,
  gammas,
  deltas,
  size = 1000
)
```

Arguments

`weights`	Numeric vector of mixture weights.
`alphas`	Numeric vector of alpha parameters.
`betas`	Numeric vector of beta parameters.
`gammas`	Numeric vector of gamma parameters.
`deltas`	Numeric vector of delta parameters.
`size`	Integer. Number of samples to generate.

Value

A list with:

samples Numeric vector of generated samples.

labels Integer vector indicating the component each sample came from.

`generate_mcculloch_table` *Generate McCulloch lookup table from simulated stable samples*

Description

Simulates alpha-stable samples across a grid of alpha and beta values, and computes quantile-based ratios used for McCulloch estimation. The result is a lookup table indexed by (alpha, beta) keys.

Usage

```
generate_mcculloch_table(alpha_grid, beta_grid, size = 1e+05)
```

Arguments

`alpha_grid`	Vector of alpha values to simulate.
`beta_grid`	Vector of beta values to simulate.
`size`	Number of samples per simulation.

Value

Named list of quantile ratios indexed by "alpha_beta".

`generate_mixture_data` *Simulates a mixture of alpha-stable distributions with randomly sampled parameters.*

Description

Simulates a mixture of alpha-stable distributions with randomly sampled parameters.

Usage

```
generate_mixture_data(K = 2, N = 1000)
```

Arguments

`K`	Integer. Number of mixture components.
`N`	Integer. Total number of samples to generate.

Value

A list containing:

data Numeric vector of generated samples.

params List of parameters for each component (alpha, beta, gamma, delta, pi).

`generate_synthetic_data` *Generate synthetic data from two alpha-stable components*

Description

Creates a synthetic dataset composed of two alpha-stable distributions.

Usage

```
generate_synthetic_data(n = 1000)
```

Arguments

`n`	Integer. Total number of samples to generate.

Value

Numeric vector of shuffled synthetic data.

gibbs_sampler *Gibbs sampler for Gaussian mixture model*

Description

Performs Gibbs sampling for a two-component Gaussian mixture model. Updates latent assignments, component means, variances, and mixing proportions.

Usage

```
gibbs_sampler(data, iterations = 1000)
```

Arguments

`data`	Numeric vector of observations.
`iterations`	Number of Gibbs iterations.

Value

List of sampled parameters per iteration.

grad_loglik_alpha *Log-likelihood gradient with respect to alpha*

Description

Log-likelihood gradient with respect to alpha

Usage

```
grad_loglik_alpha(alpha, beta, delta, omega, x)
```

Arguments

`alpha`	Numeric scalar. Stability parameter of the stable distribution ($0 < \alpha \leq 2$), controlling tail thickness.
`beta`	Numeric scalar. Skewness parameter of the stable distribution ($-1 \leq \beta \leq 1$).
`delta`	Numeric scalar. Location parameter of the stable distribution.
`omega`	Numeric scalar. Scale parameter of the stable distribution (omega > 0).
`x`	Numeric vector. Observations at which the integral is computed.

Value

Numeric scalar: estimated gradient (or `NA_real_` on invalid input).

grad_loglik_beta *Log-likelihood gradient with respect to beta*

Description

Log-likelihood gradient with respect to beta

Usage

```
grad_loglik_beta(alpha, beta, delta, omega, x)
```

Arguments

alpha	Numeric scalar. Stability parameter of the stable distribution ($0 < \alpha \leq 2$), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ($-1 \leq \beta \leq 1$).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution (omega > 0).
x	Numeric vector. Observations at which the integral is computed.

Value

Numeric scalar: estimated gradient (or `NA_real_` on invalid input).

grad_loglik_delta *Log-likelihood gradient with respect to delta (scale)*

Description

Log-likelihood gradient with respect to delta (scale)

Usage

```
grad_loglik_delta(alpha, beta, delta, omega, x)
```

Arguments

alpha	Numeric scalar. Stability parameter of the stable distribution ($0 < \alpha \leq 2$), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ($-1 \leq \beta \leq 1$).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution (omega > 0).
x	Numeric vector. Observations at which the integral is computed.

Value

Numeric scalar: estimated gradient (or `NA_real_` on invalid input).

`grad_loglik_omega` *Log-likelihood gradient with respect to omega (location)*

Description

Log-likelihood gradient with respect to omega (location)

Usage

```
grad_loglik_omega(alpha, beta, delta, omega, x)
```

Arguments

`alpha`	Numeric scalar. Stability parameter of the stable distribution ($0 < \alpha \leq 2$), controlling tail thickness.
`beta`	Numeric scalar. Skewness parameter of the stable distribution ($-1 \leq \beta \leq 1$).
`delta`	Numeric scalar. Location parameter of the stable distribution.
`omega`	Numeric scalar. Scale parameter of the stable distribution (omega > 0).
`x`	Numeric vector. Observations at which the integral is computed.

Value

Numeric scalar: estimated gradient (or `NA_real_` on invalid input).

`Im` *Imaginary part of the ECF integral*

Description

Imaginary part of the ECF integral

Usage

```
Im(r, u, x, bn)
```

Arguments

`r`	Integration variable.
`u`	Frequency.
`x`	Data point.
`bn`	Bandwidth.

Value

Imaginary component value.

`integrate_cosine` *Integrate cosine exponential*

Description

Computes the integral of the cosine exponential function for stable distributions.

Usage

```
integrate_cosine(x, alpha, beta, delta, omega)
```

Arguments

`x`	Numeric vector. Observations at which the integral is computed.
`alpha`	Numeric scalar. Stability parameter of the stable distribution ($0 < \alpha \leq 2$), controlling tail thickness.
`beta`	Numeric scalar. Skewness parameter of the stable distribution ($-1 \leq \beta \leq 1$).
`delta`	Numeric scalar. Location parameter of the stable distribution.
`omega`	Numeric scalar. Scale parameter of the stable distribution (omega > 0).

Value

Numeric vector of integral values.

`integrate_cosine_log_weighted`
Integrate cosine-log-weighted exponential

Description

Computes the integral of the cosine-log-weighted exponential function for stable distributions.

Usage

```
integrate_cosine_log_weighted(x, alpha, beta, delta, omega)
```

Arguments

x	Numeric vector. Observations at which the integral is computed.
alpha	Numeric scalar. Stability parameter of the stable distribution ($0 < \alpha \leq 2$), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ($-1 \leq \beta \leq 1$).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution (omega > 0).

Value

Numeric vector of integral values.

`integrate_function` *Robust integration helper function*

Description

Applies ‘safe_integrate‘ to a given integrand over a vector of x values.

Usage

```
integrate_function(f, x, alpha, beta, delta, omega, upper = 50, eps = 1e-08)
```

Arguments

f	Function. The integrand function to be evaluated.
x	Numeric vector. Observations at which the integral is computed.
alpha	Numeric scalar. Stability parameter of the stable distribution ($0 < \alpha \leq 2$), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ($-1 \leq \beta \leq 1$).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution (omega > 0).
upper	Numeric scalar. Upper limit of integration (default = 50).
eps	Numeric scalar. Tolerance for numerical integration (default = 1e-8).

Value

Numeric vector of integral values evaluated at each element of `x`.

integrate_sine *Integrate sin exponential*

Description

Computes the integral of the sin exponential function for stable distributions.

Usage

```
integrate_sine(x, alpha, beta, delta, omega)
```

Arguments

x	Numeric vector. Observations at which the integral is computed.
alpha	Numeric scalar. Stability parameter of the stable distribution ($0 < \alpha \leq 2$), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ($-1 \leq \beta \leq 1$).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution (omega > 0).

Value

Numeric vector of integral values.

integrate_sine_log_weighted *Integrate sine-log-weighted exponential*

Description

Computes the integral of the sine-log-weighted exponential function for stable distributions.

Usage

```
integrate_sine_log_weighted(x, alpha, beta, delta, omega)
```

Arguments

x	Numeric vector. Observations at which the integral is computed.
alpha	Numeric scalar. Stability parameter of the stable distribution ($0 < \alpha \leq 2$), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ($-1 \leq \beta \leq 1$).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution (omega > 0).

Value

Numeric vector of integral values.

---

integrate_sine_r_weighted *Integrate sine-r-weighted exponential*

Description

Computes the integral of the sine-r-weighted exponential function for stable distributions.

Usage

```
integrate_sine_r_weighted(x, alpha, beta, delta, omega)
```

Arguments

`x`	Numeric vector. Observations at which the integral is computed.
`alpha`	Numeric scalar. Stability parameter of the stable distribution ($0 < \alpha \leq 2$), controlling tail thickness.
`beta`	Numeric scalar. Skewness parameter of the stable distribution ($-1 \leq \beta \leq 1$).
`delta`	Numeric scalar. Location parameter of the stable distribution.
`omega`	Numeric scalar. Scale parameter of the stable distribution (omega > 0).

Value

Numeric vector of integral values.

---

integrate_sine_weighted *Integration wrappers for specific integrands*

Description

Integration wrappers for specific integrands

Usage

```
integrate_sine_weighted(x, alpha, beta, delta, omega)
```

Arguments

x	Numeric vector. Observations at which the integral is computed.
alpha	Numeric scalar. Stability parameter of the stable distribution ($0 < \alpha \leq 2$), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ($-1 \leq \beta \leq 1$).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution (omega > 0).

Value

Numeric vector of integral values.

Int_Im *Integrate imaginary component over $\mathbb{R}$*

Description

Integrate imaginary component over $\mathbb{R}$

Usage

```
Int_Im(u, x, bn)
```

Arguments

u	Frequency.
x	Data point.
bn	Bandwidth.

Value

Integrated imaginary value.

Int_Re   *Integrate real component over $\mathbb{R}$*

Description

Integrate real component over $\mathbb{R}$

Usage

```
Int_Re(u, x, bn)
```

Arguments

u	Frequency.
x	Data point.
bn	Bandwidth.

Value

Integrated real value.

kde_bandwidth_plugin   *KDE bandwidth selection using plugin method*

Description

KDE bandwidth selection using plugin method

Usage

```
kde_bandwidth_plugin(X, alpha)
```

Arguments

X	Numeric vector of data.
alpha	Stability parameter.

Value

Bandwidth value.

log_likelihood_mixture *Log-likelihood for mixture of stable distributions*

Description

Log-likelihood for mixture of stable distributions

Usage

```
log_likelihood_mixture(params, data)
```

Arguments

params	Numeric vector of parameters.
data	Numeric vector of observations.

Value

Negative log-likelihood (for minimization).

L_stable *Negative log-likelihood for stable distribution using dstable*

Description

Computes the negative log-likelihood of a stable distribution given parameters and data.

Usage

```
L_stable(param, obs)
```

Arguments

param	Numeric vector of parameters: alpha, beta, gamma, delta.
obs	Numeric vector of observations.

Value

Scalar value of negative log-likelihood.

Max_vrai — *Maximum likelihood estimation using Nelder-Mead*

Description

Estimates parameters of a stable distribution using the Nelder-Mead method with penalty constraints.

Usage

```
Max_vrai(x)
```

Arguments

x	Numeric vector of observations.

Value

List with estimated alpha, beta, gamma, delta.

mcculloch_lookup_estimate — *Estimate stable parameters using McCulloch lookup*

Description

Estimates alpha and beta using quantile ratios and interpolation functions. Gamma and delta are derived directly from quantiles.

Usage

```
mcculloch_lookup_estimate(
  X,
  interpolators = NULL,
  interp_alpha = NULL,
  interp_beta = NULL
)
```

Arguments

X	Numeric vector of data.
interpolators	Optional list with interp_alpha and interp_beta functions.
interp_alpha	Optional function to interpolate alpha.
interp_beta	Optional function to interpolate beta.

Value

List with estimated alpha, beta, gamma, delta.

`mcculloch_quantile_init` *Initialization using McCulloch quantile method*

Description

Estimates all four stable parameters using quantile ratios and mock lookup. Useful for initializing optimization algorithms.

Usage

```
mcculloch_quantile_init(X)
```

Arguments

`X`	Numeric vector of data.

Value

List with estimated alpha, beta, gamma, delta.

`metropolis_hastings` *Metropolis-Hastings MCMC for stable mixture clustering*

Description

Performs Metropolis-Hastings sampling to estimate parameters of a two-component alpha-stable mixture model. Proposals are generated for each parameter and accepted based on likelihood ratios. Cluster assignments are updated at each iteration.

Usage

```
metropolis_hastings(fct, iterations, lok, aa = c(1, 1), proposal_std = 0.1)
```

Arguments

`fct`	Function to estimate initial parameters.
`iterations`	Number of MCMC iterations.
`lok`	Numeric vector of observations.
`aa`	Prior parameters for Dirichlet distribution.
`proposal_std`	Standard deviation for proposal distributions.

Value

List of sampled parameters and weights across iterations.

---

mixture_stable_pdf — *Mixture of two stable PDFs*

Description

Mixture of two stable PDFs

Usage

```
mixture_stable_pdf(x, p1, p2, w)
```

Arguments

x	Numeric vector.
p1	List of parameters for first stable distribution.
p2	List of parameters for second stable distribution.
w	Mixture weight ($0 < w < 1$).

Value

Numeric vector of mixture PDF values.

---

mle_estimate — *Simple MLE estimation with default starting values*

Description

Estimates stable distribution parameters using Nelder-Mead optimization from default or user-provided starting values.

Usage

```
mle_estimate(X, x0 = NULL)
```

Arguments

X	Numeric vector of observations.
x0	Optional starting values.

Value

List with estimated alpha, beta, gamma, delta.

mock_gibbs_sampling *Mock Gibbs sampling for alpha-stable mixture estimation*

Description

Performs a simplified Gibbs sampling procedure to estimate parameters of a two-component alpha-stable mixture. Samples are drawn from prior distributions and evaluated using log-likelihood. The best parameter set is selected based on likelihood.

Usage

```
mock_gibbs_sampling(data, n_samples = 500, verbose = FALSE)
```

Arguments

`data`	Numeric vector of observations.
`n_samples`	Number of Gibbs samples to draw.
`verbose`	Logical, whether to print best log-likelihood.

Value

List containing best_params and all sampled parameter sets.

mock_lookup_alpha_beta *Mock lookup for alpha and beta (fallback)*

Description

Provides a fallback estimation of alpha and beta from quantile ratios using simple linear approximations.

Usage

```
mock_lookup_alpha_beta(v_alpha, v_beta)
```

Arguments

`v_alpha`	Quantile-based shape ratio.
`v_beta`	Quantile-based skewness ratio.

Value

List with estimated alpha and beta.

negative_log_likelihood — *Negative log-likelihood for single stable distribution*

Description

Negative log-likelihood for single stable distribution

Usage

```
negative_log_likelihood(params, data)
```

Arguments

`params`	Numeric vector (alpha, beta, gamma, delta).
`data`	Numeric vector of observations.

Value

Negative log-likelihood.

normalized_grad_alpha — *Normalized gradient for alpha parameter Computes the normalized gradient of the log-likelihood with respect to the alpha parameter over a set of observations. This is useful for optimization routines where scale-invariant updates are preferred.*

Description

Normalized gradient for alpha parameter Computes the normalized gradient of the log-likelihood with respect to the alpha parameter over a set of observations. This is useful for optimization routines where scale-invariant updates are preferred.

Usage

```
normalized_grad_alpha(alpha, beta, delta, omega, x)
```

Arguments

`alpha`	Stability parameter.
`beta`	Skewness parameter.
`delta`	Scale parameter.
`omega`	Location parameter.
`x`	Numeric vector of observations.

Value

Scalar value representing the normalized gradient.

---

normalized_objective_beta

Normalized objective for beta parameter

Description

Computes the normalized gradient of the log-likelihood with respect to the beta parameter using the latest values of alpha, delta, and omega stored in global vectors.

Usage

```
normalized_objective_beta(beta, X1, L_alpha, L_delta, L_omega)
```

Arguments

`beta`	Skewness parameter.
`X1`	Numeric vector of observations.
`L_alpha`	Vector of alpha values (global memory).
`L_delta`	Vector of delta values (global memory).
`L_omega`	Vector of omega values (global memory).

Value

Scalar value representing the normalized gradient.

---

normalized_objective_delta

Normalized objective for delta parameter

Description

Computes the normalized gradient of the log-likelihood with respect to the delta parameter using the latest values of alpha, beta, and omega stored in global vectors.

Usage

```
normalized_objective_delta(delta, X1, L_alpha, L_beta, L_omega)
```

Arguments

`delta`	Location parameter.
`X1`	Numeric vector of observations.
`L_alpha`	Vector of alpha values.
`L_beta`	Vector of beta values.
`L_omega`	Vector of omega values.

Value

Scalar value representing the normalized gra

`normalized_objective_omega` *Normalized objective for omega parameter*

Description

Computes the normalized gradient of the log-likelihood with respect to the omega parameter using the latest values of alpha, beta, and delta stored in global vectors.

Usage

```
normalized_objective_omega(omega, X1, L_alpha, L_beta, L_delta)
```

Arguments

`omega`	Location parameter.
`X1`	Numeric vector of observations.
`L_alpha`	Vector of alpha values.
`L_beta`	Vector of beta values.
`L_delta`	Vector of delta values.

Value

Scalar value representing the normalized gradient.

`N_epanechnikov` *Epanechnikov kernel*

Description

Epanechnikov kernel

Usage

```
N_epanechnikov(z)
```

Arguments

`z`	Numeric input.

Value

Kernel value.

`N_gaussian` *Gaussian kernel*

Description

Gaussian kernel

Usage

```
N_gaussian(z)
```

Arguments

`z`	Numeric input.

Value

Kernel value.

N_uniform *Uniform kernel*

Description

Uniform kernel

Usage

```
N_uniform(z)
```

Arguments

z	Numeric input.

Value

Kernel value.

plot_comparison *Compare EM-estimated mixture with a non-optimized reference model*

Description

Plots the fitted EM mixture density alongside a manually defined reference mixture to visually compare estimation quality.

Usage

```
plot_comparison(data, p1, p2, w)
```

Arguments

data	Numeric vector of observations.
p1	Vector of parameters for component 1 (alpha, beta, gamma, delta).
p2	Vector of parameters for component 2.
w	Mixing weight for component 1.

Value

Invisibly returns the file path to the saved PNG image of the comparison plot.

plot_distributions *Plot histogram with normal and stable PDF overlays*

Description

Displays a histogram of the data with overlaid normal and alpha-stable probability density functions.

Usage

```
plot_distributions(x, params_stable)
```

Arguments

`x`	Numeric vector of data.
`params_stable`	List with alpha, beta, gamma, delta.

Value

NULL (plot is displayed as a side effect)

plot_effective_reproduction_number *Plot effective reproduction number (Re) over time*

Description

Computes and visualizes the effective reproduction number (Rt) over time using incidence data and generation time distribution. Optionally overlays a smoothed spline curve.

Usage

```
plot_effective_reproduction_number(
  GT,
  S,
  inc,
  dates,
  est_r0_ml,
  RT,
  output_file = "RN_avec_dates_EM-ML.pdf"
)
```

Arguments

`GT`	Numeric vector. Generation time distribution.
`S`	Numeric vector. Secondary cases.
`inc`	Numeric vector. Incidence time series.
`dates`	Date vector corresponding to observations.
`est_r0_ml`	Function to estimate R0.
`RT`	Function to compute Rt.
`output_file`	Output PDF filename.

Value

NULL (plot is saved as PDF)

`plot_final_mixture_fit` *Plot final fitted mixture of alpha-stable distributions*

Description

Plots the final fitted mixture PDF over the data histogram.

Usage

```
plot_final_mixture_fit(data, p1, p2, w)
```

Arguments

`data`	Numeric vector of observations.
`p1`	Vector of parameters for component 1.
`p2`	Vector of parameters for component 2.
`w`	Mixing weight for component 1.

Value

NULL (plot is saved as PNG)

`plot_fit_vs_true` *Plot true vs estimated mixture density*

Description

Compares the true mixture density with the estimated one by overlaying both on the histogram of the observed data.

Usage

```
plot_fit_vs_true(true_params, est_params, data, bins = 100)
```

Arguments

`true_params`	List of true mixture components with alpha, beta, gamma, delta, pi.
`est_params`	List of estimated mixture components.
`data`	Numeric vector of observations.
`bins`	Number of histogram bins.

Value

NULL (plot is displayed)

`plot_fit_vs_true_methods` *Compare estimated mixture densities from two methods against the true density*

Description

Plots the true mixture density and compares it with two estimated densities obtained from different methods (e.g., MLE vs ECF).

Usage

```
plot_fit_vs_true_methods(
  data,
  true_params,
  method1 = "MLE",
  method2 = "ECF",
  bins = 100
)
```

Arguments

`data`	Numeric vector of observations.
`true_params`	List of true mixture components.
`method1`	First estimation method ("MLE", "ECF", etc.).
`method2`	Second estimation method.
`bins`	Number of histogram bins.

Value

NULL (plot is displayed)

`plot_method_comparison` *Plot RMSE and Log-Likelihood comparison across methods*

Description

Generates bar plots comparing RMSE and log-likelihood values across different estimation methods.

Usage

```
plot_method_comparison(scores)
```

Arguments

`scores`	Named list of score objects with RMSE and LogLikelihood.

Value

NULL (plot is saved as PNG)

`plot_mixture` *Plot mixture of two alpha-stable distributions*

Description

Plots the histogram of data and overlays the estimated mixture density and its components.

Usage

```
plot_mixture(data, params1, params2, w, label = "EM")
```

Arguments

data	Numeric vector of observations.
params1	Vector of parameters for component 1 (alpha, beta, gamma, delta).
params2	Vector of parameters for component 2.
w	Mixing weight for component 1.
label	Optional label for output file.

Value

NULL (plot is saved as PNG)

---

plot_mixture_fit *Plot mixture fit with individual components*

Description

Displays the estimated mixture density and optionally its individual components over a histogram of the data.

Usage

```
plot_mixture_fit(
  data,
  estimated_params,
  bins = 200,
  plot_components = TRUE,
  save_path = NULL,
  show_plot = TRUE,
  title = "Mixture of Alpha-Stable Distributions"
)
```

Arguments

data	Numeric vector of observations.
estimated_params	List with weights, alphas, betas, gammas, deltas.
bins	Number of histogram bins.
plot_components	Logical, whether to show individual components.
save_path	Optional PNG filename.
show_plot	Logical, whether to display the plot.
title	Plot title.

Value

Invisibly returns the file path to the saved plot (if `save_path` is provided), or `NULL` otherwise.

plot_real_mixture_fit *Plot fitted mixture on real dataset*

Description

Plots the fitted mixture model over a histogram of real-world data, showing both components and the overall mixture.

Usage

```
plot_real_mixture_fit(X, result)
```

Arguments

X	Numeric vector of observations.
result	List containing params1, params2, and lambda1.

Value

NULL (plot is saved as PNG)

plot_results *Plot posterior mixture density from MCMC samples*

Description

Visualizes the estimated two-component alpha-stable mixture density using parameters obtained from MCMC sampling. Optionally overlays the true density for comparison.

Usage

```
plot_results(
  M2_w1,
  M2_alpha1,
  M2_beta1,
  M2_delta1,
  M2_omega1,
  M2_w2,
  M2_alpha2,
  M2_beta2,
  M2_delta2,
  M2_omega2,
  xx,
  xx_true = NULL,
  yy_true = NULL
)
```

Arguments

`M2_w1`	Numeric scalar. Mixture weight of first component.
`M2_alpha1`	Numeric vector. Stability parameter samples of first component.
`M2_beta1`	Numeric vector. Skewness parameter samples of first component.
`M2_delta1`	Numeric vector. Location parameter samples of first component.
`M2_omega1`	Numeric vector. Scale parameter samples of first component.
`M2_w2`	Numeric scalar. Mixture weight of second component.
`M2_alpha2`	Numeric vector. Stability parameter samples of second component.
`M2_beta2`	Numeric vector. Skewness parameter samples of second component.
`M2_delta2`	Numeric vector. Location parameter samples of second component.
`M2_omega2`	Numeric vector. Scale parameter samples of second component.
`xx`	Numeric vector of grid values for density evaluation.
`xx_true`	Optional numeric vector for true density x-values.
`yy_true`	Optional numeric vector for true density y-values.

Value

Invisibly returns the file path to the saved PNG image of the posterior mixture density plot.

`plot_trace` *Plot trace of a parameter across MCMC iterations*

Description

Displays the evolution of a parameter across MCMC iterations to assess convergence.

Usage

```
plot_trace(samples, param_name)
```

Arguments

`samples`	List of sample objects from MCMC.
`param_name`	Name of the parameter to trace.

Value

NULL (plot is displayed as a side effect)

plot_vs_normal_stable *Plot comparison between normal and stable distributions*

Description

Displays a histogram of the data with overlaid normal and alpha-stable PDFs.

Usage

```
plot_vs_normal_stable(x, params_stable, fig_path = NULL)
```

Arguments

x	Numeric vector of data.
params_stable	List with alpha, beta, gamma, delta.
fig_path	Optional path to save the plot (PNG). If NULL, uses tempdir().

Value

NULL (saves plot to file)

qcv_stat *QCV statistic for tail heaviness*

Description

Computes the QCV (Quantile Conditional Variance) statistic to assess tail heaviness of a distribution.

Usage

```
qcv_stat(x)
```

Arguments

x	Numeric vector of data.

Value

Numeric: QCV value.

Re — *Real part of the ECF integral*

Description

Real part of the ECF integral

Usage

```
Re(r, u, x, bn)
```

Arguments

r	Integration variable.
u	Frequency.
x	Data point.
bn	Bandwidth.

Value

Real component value.

recursive_weight — *Recursive weight function*

Description

Recursive weight function

Usage

```
recursive_weight(l)
```

Arguments

l	Index.

Value

Weight value.

robust_ecf_regression *Estimate stable parameters using robust ECF regression*

Description

Estimate stable parameters using robust ECF regression

Usage

```
robust_ecf_regression(x, u)
```

Arguments

`x`	Numeric vector of data.
`u`	Vector of frequency values.

Value

A list with estimated parameters: `alpha`, `beta`, `gamma`, and `delta`.

robust_mle_estimate *Robust MLE estimation with multiple starting points*

Description

Performs multiple MLE estimations with randomized starting points and selects the best result based on log-likelihood.

Usage

```
robust_mle_estimate(data, n_starts = 5)
```

Arguments

`data`	Numeric vector of observations.
`n_starts`	Number of random initializations.

Value

List with best estimated alpha, beta, gamma, delta.

rstable	*Generate random samples from stable distribution*

Description

Generate random samples from stable distribution

Usage

```
rstable(n, alpha, beta, gamma = 1, delta = 0, pm = 1)
```

Arguments

`n`	Number of samples.
`alpha`	Stability parameter (0,2].
`beta`	Skewness parameter [-1,1].
`gamma`	Scale (>0).
`delta`	Location.
`pm`	Parameterization (0 or 1).

Value

Numeric vector of samples.

RT	*Compute effective reproduction number Rt*

Description

Estimates the effective reproduction number (Rt) over time using incidence data and a generation time distribution.

Usage

```
RT(inc, G)
```

Arguments

`inc`	Numeric vector of incidence counts.
`G`	Numeric vector representing the generation time distribution.

Value

Numeric vector of Rt values.

run_all_estimations *Run all EM-based estimations without Gibbs sampling (CRAN-safe)*

Description

Executes multiple EM algorithms (recursive ECF, kernel ECF, weighted OLS, CDF-based) on the input data. Optionally saves mixture plots to a temporary directory.

Usage

```
run_all_estimations(X1, bw_sj, max_iter = 200, tol = 1e-04, save_plots = FALSE)
```

Arguments

X1	Numeric vector of data.
bw_sj	Numeric. Bandwidth value (e.g., Silverman's rule).
max_iter	Integer. Maximum number of EM iterations.
tol	Numeric. Convergence tolerance.
save_plots	Logical. Whether to save PNG plots to tempdir().

Value

Invisibly returns a list containing fitted parameters for each EM method.

run_estimations_with_gibbs *Run all EM-based estimations with Gibbs sampling (CRAN-safe)*

Description

Executes EM algorithms with Gibbs sampling on the input data. Optionally saves mixture plots to a temporary directory.

Usage

```
run_estimations_with_gibbs(
  data,
  bw_sj,
  max_iter = 100,
  tol = 1e-04,
  save_plots = FALSE
)
```

Arguments

`data`	Numeric vector of observations.
`bw_sj`	Numeric. Bandwidth value (e.g., Silverman's rule).
`max_iter`	Integer. Maximum number of EM iterations.
`tol`	Numeric. Convergence tolerance.
`save_plots`	Logical. Whether to save PNG plots to tempdir().

Value

Invisibly returns a list containing fitted parameters for each EM method with Gibbs.

`r_stable_pdf` *Robust stable PDF computation*

Description

Robust stable PDF computation

Usage

```
r_stable_pdf(x, alpha, beta, scale, location)
```

Arguments

`x`	Points at which to evaluate the density.
`alpha`	Stability parameter (0,2].
`beta`	Skewness parameter [-1,1].
`scale`	Scale (>0).
`location`	Location parameter.

Value

Numeric vector of density values.

safe_integrate *Safe integration wrapper with multiple fallback strategies*

Description

Performs numerical integration with progressively more conservative settings to improve robustness. Useful when standard integration may fail due to oscillatory or heavy-tailed functions.

Usage

```
safe_integrate(
  integrand_func,
  lower = 1e-08,
  upper = 50,
  max_attempts = 3,
  ...
)
```

Arguments

integrand_func	Function to integrate. Must accept a numeric vector as input and return numeric values.
lower	Numeric scalar. Lower bound of the integration interval (default = 1e-8 to avoid singularities at 0).
upper	Numeric scalar. Upper bound of the integration interval (default = 50).
max_attempts	Integer. Maximum number of fallback attempts with modified settings if the initial integration fails.
...	Additional arguments passed to `integrand_func`.

Value

Numeric scalar. The estimated value of the integral, or 0 if all attempts fail.

simple_em_real *Simple 2-component EM using ECF initialization*

Description

Initializes EM using k-means clustering and ECF-based parameter estimation.

Usage

```
simple_em_real(X, max_iter = 10)
```

Arguments

`X`	Numeric vector of observations.
`max_iter`	Integer. Maximum number of iterations.

Value

List with lambda1 and estimated parameters for both components.

`simulate_mixture` *Simulate mixture data from alpha-stable components*

Description

Generates synthetic data from a mixture of alpha-stable distributions using specified weights and parameters.

Usage

```
simulate_mixture(n, weights, params)
```

Arguments

`n`	Number of samples to generate.
`weights`	Vector of mixture weights.
`params`	List of parameter sets for each component (each with alpha, beta, gamma, delta).

Value

Numeric vector of simulated samples.

`sine_exp_ralpha` *Sine exponential function*

Description

Sine exponential function

Usage

```
sine_exp_ralpha(r, x, alpha, beta, delta, omega)
```

Arguments

r	Integration variable.
x	Observation.
alpha	Numeric scalar. Stability parameter of the stable distribution ($0 < \alpha \leq 2$), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ($-1 \leq \beta \leq 1$).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution (omega > 0).

Value

Numeric value of the integrand.

---

sine_log_weighted_exp_ralpha

Sine-log-weighted exponential with r^(-alpha) term

Description

Sine-log-weighted exponential with r^(-alpha) term

Usage

```
sine_log_weighted_exp_ralpha(r, x, alpha, beta, delta, omega)
```

Arguments

r	Integration variable.
x	Observation.
alpha	Numeric scalar. Stability parameter of the stable distribution ($0 < \alpha \leq 2$), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ($-1 \leq \beta \leq 1$).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution (omega > 0).

Value

Numeric value of the integrand.

`sine_r_weighted_exp_ralpha`

Sine-r-weighted exponential function

Description

Sine-r-weighted exponential function

Usage

```
sine_r_weighted_exp_ralpha(r, x, alpha, beta, delta, omega)
```

Arguments

`r`	Integration variable.
`x`	Observation.
`alpha`	Numeric scalar. Stability parameter of the stable distribution ($0 < \alpha \leq 2$), controlling tail thickness.
`beta`	Numeric scalar. Skewness parameter of the stable distribution ($-1 \leq \beta \leq 1$).
`delta`	Numeric scalar. Location parameter of the stable distribution.
`omega`	Numeric scalar. Scale parameter of the stable distribution (omega > 0).

Value

Numeric value of the integrand.

`sine_weighted_exp_ralpha`

Sine-weighted exponential with r^alpha term

Description

Sine-weighted exponential with r^alpha term

Usage

```
sine_weighted_exp_ralpha(r, x, alpha, beta, delta, omega)
```

Arguments

r	Integration variable.
x	Observation.
alpha	Numeric scalar. Stability parameter of the stable distribution (($0 < \alpha \leq 2$), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ($-1 \leq \beta \leq 1$).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution (omega > 0).

Value

Numeric value of the integrand.

---

skew_kurtosis *Calculate skewness and kurtosis*

Description

Computes the skewness and kurtosis of a numeric vector.

Usage

```
skew_kurtosis(x)
```

Arguments

x	Numeric vector of data.

Value

List with skewness and kurtosis.

---

stable_fit_init *Initialize stable distribution parameters*

Description

Initialize stable distribution parameters

Usage

```
stable_fit_init(x)
```

Arguments

x Numeric vector of data.

Value

A list with estimated parameters (alpha, beta, gamma, delta).

TableS2_serial_interval_mean_ *Example transmission pair data with mean serial interval*

Description

A data frame of infector-infectee pairs with demographic and serial interval information.

Usage

```
data(TableS2_serial_interval_mean_)
```

Format

A data frame with columns:

infector_id Character. Infector ID.

infector_age Numeric. Age of infector.

infector_sex Character. Sex of infector.

infector_onsetDate Date. Infector symptom onset.

infector_labConfirmDate Date. Infector lab confirmation.

infectee_id Character. Infectee ID.

infectee_age Numeric. Age of infectee.

infectee_sex Character. Sex of infectee.

infectee_onsetDate Date. Infectee symptom onset.

infectee_labConfirmDate Date. Infectee lab confirmation.

serial_interval_mean_based Numeric. Mean-based serial interval.

Source

Simulated or real data for demonstration.

test_normality *Test normality using multiple statistical tests*

Description

Applies Shapiro-Wilk, Jarque-Bera, Anderson-Darling, and Kolmogorov-Smirnov tests to assess normality of a dataset.

Usage

```
test_normality(x)
```

Arguments

x	Numeric vector of data.

Value

List of test statistics and p-values.

unpack_params *Helper function to unpack parameters*

Description

Helper function to unpack parameters

Usage

```
unpack_params(p)
```

Arguments

p	A list containing alpha, beta, gamma, delta.

Value

A numeric vector of parameters.

validate_params *Validate and clip parameters for stable distribution*

Description

Ensures parameters are within valid bounds for stable distribution computations.

Usage

```
validate_params(alpha, beta, delta, r = NULL)
```

Arguments

alpha	Stability parameter (must be > 0 and < 2).
beta	Skewness parameter (clipped to [-1, 1]).
delta	Scale parameter (must be > 0).
r	Optional numeric vector. Must be positive if provided.

Value

Clipped beta value.

wasserstein_distance_mixture *Wasserstein distance between two mixture distributions*

Description

Wasserstein distance between two mixture distributions

Usage

```
wasserstein_distance_mixture(params1, params2, size = 5000)
```

Arguments

params1	List of parameters for first mixture.
params2	List of parameters for second mixture.
size	Number of samples to approximate distance.

Value

Wasserstein distance.

Index